

Article

A Proof-of-Concept FPGA-Based Clock Signal Phase Alignment System

Andrzej A. Wojciechowski 

Institute of Microelectronics and Optoelectronics, Warsaw University of Technology, ul. Koszykowa 75, 00-662 Warsaw, Poland; andrzej.wojciechowski2.dokt@pw.edu.pl

Abstract: Phase alignment of periodic events between multiple systems is required in multiple fields and applications. Most of the existing solutions focus on either low frequency and relatively low accuracy or high complexity, high accuracy and precision. In contrast, this work aimed to develop an intermediate solution, supporting high frequencies and relatively high accuracy and precision, with relatively low complexity. A hypothetical concept and mathematical model is presented with a hardware test implementation based entirely on FPGA resources. Deliberate resource selection and utilization enables a significant simplification of calculations and, as a result, a reduction in logic resource utilization. The proposed concept was implemented and verified using the AMD/Xilinx Artix 7 35T FPGA platform.

Keywords: field-programmable gate array (FPGA); phase alignment; phase synchronization; clock synchronization; daisy chain; tapped delay line (TDL); delay matching; clocking resources



Citation: Wojciechowski, A.A. A Proof-of-Concept FPGA-Based Clock Signal Phase Alignment System. *Electronics* **2024**, *13*, 3295. <https://doi.org/10.3390/electronics13163295>

Academic Editors: Tao Li, Zonghui Li and Yangdong Deng

Received: 31 May 2024

Revised: 30 June 2024

Accepted: 4 July 2024

Published: 20 August 2024



Copyright: © 2024 by the author. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

The importance of the synchronization of multiple individual systems increases with the demand for higher precision, reliability, and performance in electronics. Two primary kinds of synchronization models can be distinguished: time synchronization and phase alignment (or phase synchronization). The former focuses on high-level clock synchronization, usually in the range of milliseconds to tens of nanoseconds. Protocols such as NTP or PTP (IEEE 1588–2008 and prior) have been designed for this type of synchronization. The latter focuses on phase alignment of periodic events, such as the synchronization of multiple signals generated separately in multiple distinct systems. Some projects combine these two types to achieve up to sub-nanosecond accuracy and picosecond precision of synchronization [1,2].

Phase alignment between multiple systems is widely utilized in multiple applications, including acoustic tracking for accurate time-of-flight measurements [3], active sonars and AUV testing [4], power supplies for plasma materials processing [5], radar systems modeling and testing [6], 5G multiple-input multiple-output (MIMO) transmitters [7], physics experimental devices [8,9], phasor measurement units [10], and more. This work's motivation was to develop a relatively simple clock signal phase alignment mechanism for a system consisting of multiple individual ASIC (application-specific integrated circuit) or FPGA (field-programmable gate array) devices, using the least possible number of signals.

One of the most frequently used methods for phase synchronization is periodic re-alignment to a received synchronization pulse. Between the pulse intervals, the clocks involved continue running autonomously at the same frequency. If synchronization pulses are lost, the clocks remain untouched, maintaining a semi-synchronized state, with increasing mutual offset due to drifting crystal oscillators. After an appropriate synchronization pulse has been detected, the internal receiving timers are set to a predefined value, resetting the offsets to zero. A schematic diagram of the actual synchronization procedure is given in Figure 1. Besides being simple, this method is tolerant to missing sending pulses [3]. However, its resolution is fundamentally limited to a system clock period. The precision

can be further limited by a degraded rise or fall time of the synchronization pulse. A very similar method is used in many applications, such as National Instruments NI-TC1k Technology [11].

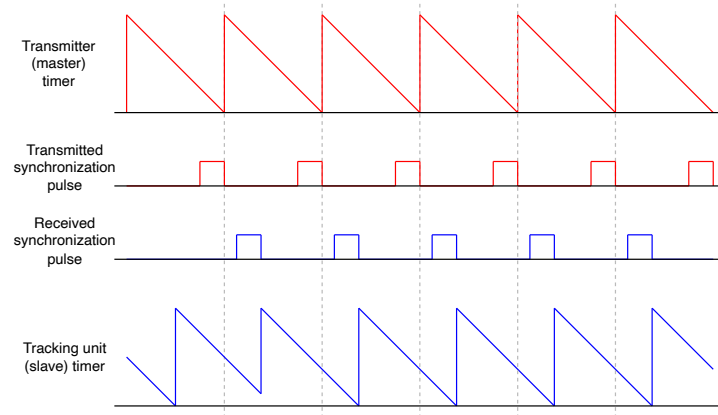


Figure 1. Concept of synchronization using period pulse realignment.

Another well-known phase alignment method is used in the white rabbit (WR) protocol. WR is an extension of the IEEE 1588 precision time protocol (PTP), initially developed to serve accelerators at the European Organization for Nuclear Research (CERN) and widely used in scientific installations. It utilizes the cooperation between Layer 1 (L1) synchronization and PTP synchronization, in order to achieve clock phase alignment and time synchronization up to sub-nanosecond accuracy and picosecond precision of synchronization [1,2]. While the white rabbit project is a good option for relatively large scale systems, at the same time it is very complex, and requires relatively large FPGA chips and additional external components, such as separate phase-locked loops (PLL) or voltage controlled oscillators (VCO). The WR network also requires the usage of dedicated white rabbit switches—the key component of any WR installation [12]. A schematic diagram of white rabbit L1 clock generation and distribution is depicted in Figure 2. As a result, it might not be the preferred solution for relatively small systems, such as synchronization of multiple chips, together forming a single device. For these types of systems, the white rabbit protocol might be excessive.

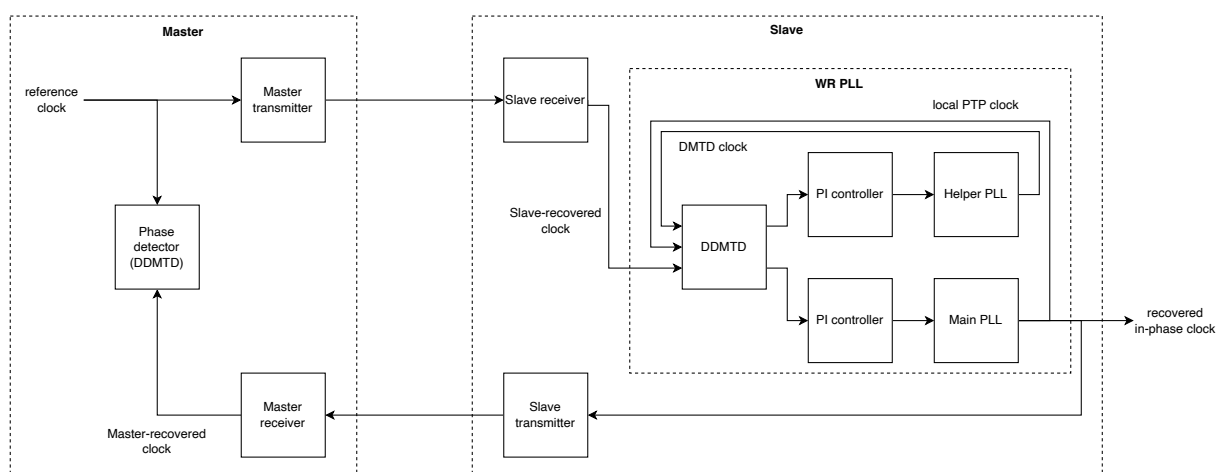


Figure 2. Simplified block diagram of clock syntonization and distribution in white rabbit (adapted from [1,12]).

In addition to the methods mentioned above, several patents have addressed the issue of phase alignment to ensure multiple system synchronization [13–15].

This paper is organized as follows: Section 2 describes the theoretical background and concept architecture of the clock signal phase alignment system. Section 3 presents a proof-of-concept hardware application using commercial FPGA development boards. The methodology of design verification and measurements is described in Section 4. The results are presented in Section 5. The paper ends with the conclusions.

2. Concept Design Architecture

The initial theoretical concept of the discussed clock signal phase alignment system was introduced in [16]. Since then, the design has been refined, implemented, and verified in hardware FPGA structure. This paper presents the developed concept with a hardware proof-of-concept implementation and results verification. This section presents the developed universal concept, followed by the FPGA-specific implementation details in Section 3.

The general block diagram of the proposed architecture is depicted in Figure 3. It includes individual delays of internal components, as well as delays of the interconnects between them on the clock synchronization paths. These delays are included in the mathematical calculations discussed subsequently. A single node is composed of the following modules:

- Phase Shifter/Delay Chain—programmable block shifting/delaying the periodic input signal by a selected value;
- Frequency Divider—selectable frequency division by 1 or 2 with delay d_{FD} ;
- Two tri-state buffers (TSB) with delays d_{TSB} ;
- Clock signal multiplexer with delay d_M ;
- Phase Comparator—block detecting the phase difference between two periodic signals;
- Digital Controller block—marked grey block in Figure 3 with respective control signals;
- Communication bus between Controller blocks—marked violet in Figure 3.

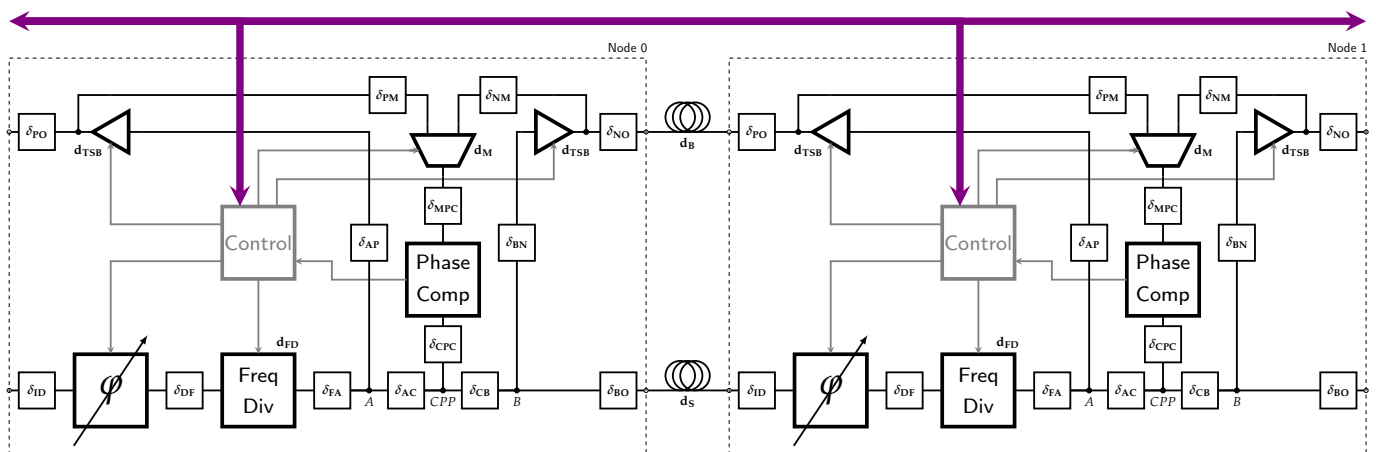


Figure 3. Concept diagram of two daisy chained clock phase alignment systems with connections and component delays marked.

The diagram also includes external connection delays marked d_S and d_B . These are unidirectional and bidirectional connections, respectively, between the two connected nodes. There are several initial assumptions that led to the development of the presented design:

- Theoretically unlimited number of systems to be phase-aligned. Only physical constraints should limit the maximum number of nodes. Figure 3 depicts only two nodes to simplify the diagram. Additional nodes can be daisy-chained as needed.
- Each system contains the same phase-alignment block.
- Isotropic propagation delays for bidirectional connections.

All of the following calculations are performed relative to the common phase point (CPP) denoted in Figure 3. However, after the calibration, all points across all interconnects connected with the CPP are phase aligned with equivalent points in other nodes. The two connected nodes are phase-aligned when the phase shifter delay D_{target} in the subsequent node compensates for the delay between the nodes' CPPs. This can be described using equation (with T marking the period of a signal for synchronization):

$$D_{target} = -(d_{CB} + d_{BO} + d_S + d_{ID} + d_{DF} + d_{FD} + d_{FA} + d_{AC}) \pmod{T} \quad (1)$$

In order to simplify the calibration, an additional constraint, affecting internal routing, as well as the FPGA resource selection or ASIC layout, had to be imposed on the internal connections delays of the presented design. Its purpose is to reduce the complexity of the equation system that is necessary to solve for phase alignment. The constraint can be described using equation:

$$d_{AP} + d_{NM} = d_{BN} + d_{AC} + d_{CB} + d_{PM} \quad (2)$$

Using an example from Figure 3, the value of phase shifter delay D can be determined using a five-stage calibration procedure:

2.1. Stage 1 of the Calibration Procedure

The first stage of the calibration procedure is executed in the subsequent node (marked as *Node 1* in Figure 3). During this stage, the clock signal is transmitted from the *Node 0* to the *Node 1* using two parallel channels: unidirectional and bidirectional, with the divisors of both frequency divider blocks set to one. The *Node 1* shifts the phase of the received signal using the phase shifter module until the phase offset of the input signals of the *Node 1* phase comparator's is at its minimum (closest to zero). The established phase shift D_1 is stored for later processing. The described calibration stage can be expressed using the following equation:

$$D_1 = d_{CB} + d_{BN} + d_{TSB} + d_{NO} + d_B + d_{PO} + d_{PM} + d_M + d_{MPC} - (d_{CB} + d_{BO} + d_S + d_{ID} + d_{DF} + d_{FD} + d_{FA} + d_{AC} + d_{CPC}) \pmod{T} \quad (3)$$

2.2. Stage 2 of the Calibration Procedure

The second stage of the calibration procedure differs from the previous stage only in the divisor value of the *Node 0* frequency divider. Therefore, calibration stage 1 is repeated using a signal with two times lower frequency compared to the original. The reason for using this two times slower clock is explained in Section 2.5. This calibration stage can be expressed using the following equation:

$$D_2 = d_{CB} + d_{BN} + d_{TSB} + d_{NO} + d_B + d_{PO} + d_{PM} + d_M + d_{MPC} - (d_{CB} + d_{BO} + d_S + d_{ID} + d_{DF} + d_{FD} + d_{FA} + d_{AC} + d_{CPC}) \pmod{2T} \quad (4)$$

2.3. Stage 3 of the Calibration Procedure

The third stage of the calibration procedure is executed partially in *Node 0* and partially in *Node 1*. During this stage the clock signal is transmitted from *Node 0* to *Node 1* using a unidirectional channel (with delay marked d_S in Figure 3) and directed back to *Node 0* using a bidirectional channel (with delay marked d_B in Figure 3). Similarly to in stage 1, the divisors of both frequency divider blocks are set to one during this calibration stage. The *Node 1* shifts the phase of a received signal using the phase shifter module until the phase offset of the input signals of the *Node 0* phase comparator is at its minimum (closest to zero). The established phase shift D_3 is stored for later processing. The described calibration stage can be expressed using the following equation:

$$D_3 = d_{CPC} - (d_{CB} + d_{BO} + d_S + d_{ID} + d_{DF} + d_{FD} + d_{FA} + d_{AP} + d_{TSB} + d_{PO} + d_B + d_{NO} + d_{NM} + d_M + d_{MPC}) \pmod{T} \quad (5)$$

2.4. Stage 4 of the Calibration Procedure

The fourth stage of the calibration procedure differs from the previous stage only in the divisor value of the *Node 0* frequency divider. Therefore, calibration stage 3 is repeated using a signal with two times lower frequency compared to the original. This calibration stage can be expressed using the following equation:

$$D_4 = d_{CPC} - (d_{CB} + d_{BO} + d_S + d_{ID} + d_{DF} + d_{FD} + d_{FA} + d_{AP} + d_{TSB} + d_{PO} + d_B + d_{NO} + d_{NM} + d_M + d_{MPC}) \pmod{2T} \quad (6)$$

2.5. Stage 5 of the Calibration Procedure—Calculating the Final Phase Shift Value

The target phase shifter value estimate is calculated by solving a system of equations constructed using Equations (2), (3), and (5). The simplified result can be expressed using the following equation:

$$D_1 + D_3 = -2(d_{AC} + d_{CB} + d_{BO} + d_S + d_{ID} + d_{DF} + d_{FD} + d_{FA}) = 2D_{target} \pmod{T} \quad (7)$$

To solve Equation (7) a modular multiplicative inverse must be calculated. Mathematically, a multiplicative inverse for a class $[a] \in \mathbb{Z}_n$ is a class $[b] \in \mathbb{Z}_n$ such that $[a][b] = [1]$. A class $[a] \in \mathbb{Z}_n$ is a unit if it has a multiplicative inverse in $\mathbb{Z}_n$. Additionally, $[a]$ is a unit in $\mathbb{Z}_n$ if and only if a and n are coprimes (if $\gcd(a, n) = 1$). As a result, if n is a composite number, then division by non-zero classes is not always possible. This can also be defined algebraically as $\mathbb{Z}_n$ being a ring, but not a field [17].

In the case of Equation (7), none of the variables can be assumed as relatively prime, and the signal period T cannot be assumed as prime (non-composite). Because of this, Equation (7) has two general results:

$$\frac{D_1 + D_3}{2} = D_{target} \pmod{T} \quad \vee \quad \frac{D_1 + D_3}{2} = D_{target} + \frac{T}{2} \pmod{T} \quad (8)$$

The two results have a 180° offset. However, it turns out that the described mathematical problem can be solved by repeating the calculations in modulo $2T$. For this, the previously mentioned results obtained for the two times slower clock signal are needed. The simplified result of the equation system constructed using Equations (2), (4) and (6) can be expressed using the following equation:

$$D_2 + D_4 = -2(d_{AC} + d_{CB} + d_{BO} + d_S + d_{ID} + d_{DF} + d_{FD} + d_{FA}) = 2D_{target} \pmod{2T} \quad (9)$$

Equation (9) can be solved similarly to Equation (7) and produces two general solutions:

$$\frac{D_2 + D_4}{2} = D_{target} \pmod{2T} \quad \vee \quad \frac{D_2 + D_4}{2} = D_{target} + T \pmod{2T} \quad (10)$$

It can be noticed that Equations (8) and (10) share one common result (up to modulo value), which corresponds to the intended value of the delay between the CPPs of the two nodes. It is worth noting that the differences between the correct result and the two false results (presented in Equations (8) and (10)) that are equal to $\frac{T}{2}$ and T should provide enough margin for inaccuracies and allow correctly determining the two phase shifter estimates with the closest values.

Depending on the implementation, satisfying the internal connection delay constraint given in Equation (2) might not be critical, e.g., when the sum of delay values is significantly less than clock period value. However, this work describes a general case, when such an assumption cannot be made. Additionally, as described in Section 3.5, the sum of internal delays included in the constraint in the implemented proof-of-concept design may correspond to the period of a 350 MHz clock signal. Clock signal frequency values of this order of magnitude are often encountered in practice.

The previously mentioned communication bus between the nodes is used to ensure that both nodes correctly transition to the required calibration stages and transmit the result of the phase comparator operations from *Node 0* to *Node 1*.

2.6. Phase Comparison

The primary function of the phase comparator block is to detect whether the two input signals of the same frequency are in phase and to indicate this to the digital controller. The information about the exact value of phase difference is not an issue at this stage. The phase comparison can be performed in multiple different ways; e.g., the charge-pump based phase detector used in PLLs [18], a stochastic frequency–phase detector [19], XOR-based phase detector [20], dual mixer time difference (DMTD) [21], or digital dual mixer time difference (D-DMTD) [22]. This work utilizes a TDL-based phase detector [23], which will be discussed further in Section 3.6. Some types of phase detectors can only be used in ASICs, and others are instead used in FPGAs. Any of the mentioned phase detectors could theoretically be implemented in the presented phase alignment system. However, the list of phase comparators that could be used in conjunction with the presented system is not limited to the ones listed in the text.

2.7. Concept Simulations

Prior to hardware proof-of-concept implementation, general mathematical simulations utilizing only functional behavioral models were conducted to find out whether the presented mathematical model is correct. The simulations were implemented and performed in GNU Octave [24] as well as Verilog HDL. Figure 4 depicts a graphical representation of an example mathematical simulation result. The diagrams depict a sum of partial delays, constituting the total delay between the CPPs, in modulo T and $2T$. An additional diagram illustrates four calculated estimates. The overlap between two of them corresponds to the actual total delay between the CPPs. The calculations were performed multiple times—each time with randomly selected values of each delay, as well as the clock period. During each of over 1000 simulations, the correct phase shift value was selected.

To further verify the concept, a digital simulation containing random interconnect and component delays was created. A behavioral model of each component was implemented, with parameterized transmission delay values. To verify the capability of the phase alignment system to synchronize multiple nodes, the simulated daisy-chain connection consisted of 10 separate instances. All instances were connected similarly to the diagram presented in Figure 3. The results further confirmed the viability of the concept, even for multiple daisy-chained systems. They also confirmed the initial conclusion that the final alignment precision depends on the phase shifter resolution, as well as the precision of the performed calculations. Another conclusion was that, theoretically, the total misalignment in a chain of multiple nodes can be reduced. The calibration algorithm could be modified to additionally use the previous nodes' calculation results. Consequently, in the worst case, the alignment error accumulation would be reduced. The behavioral simulations were described and published in [16].

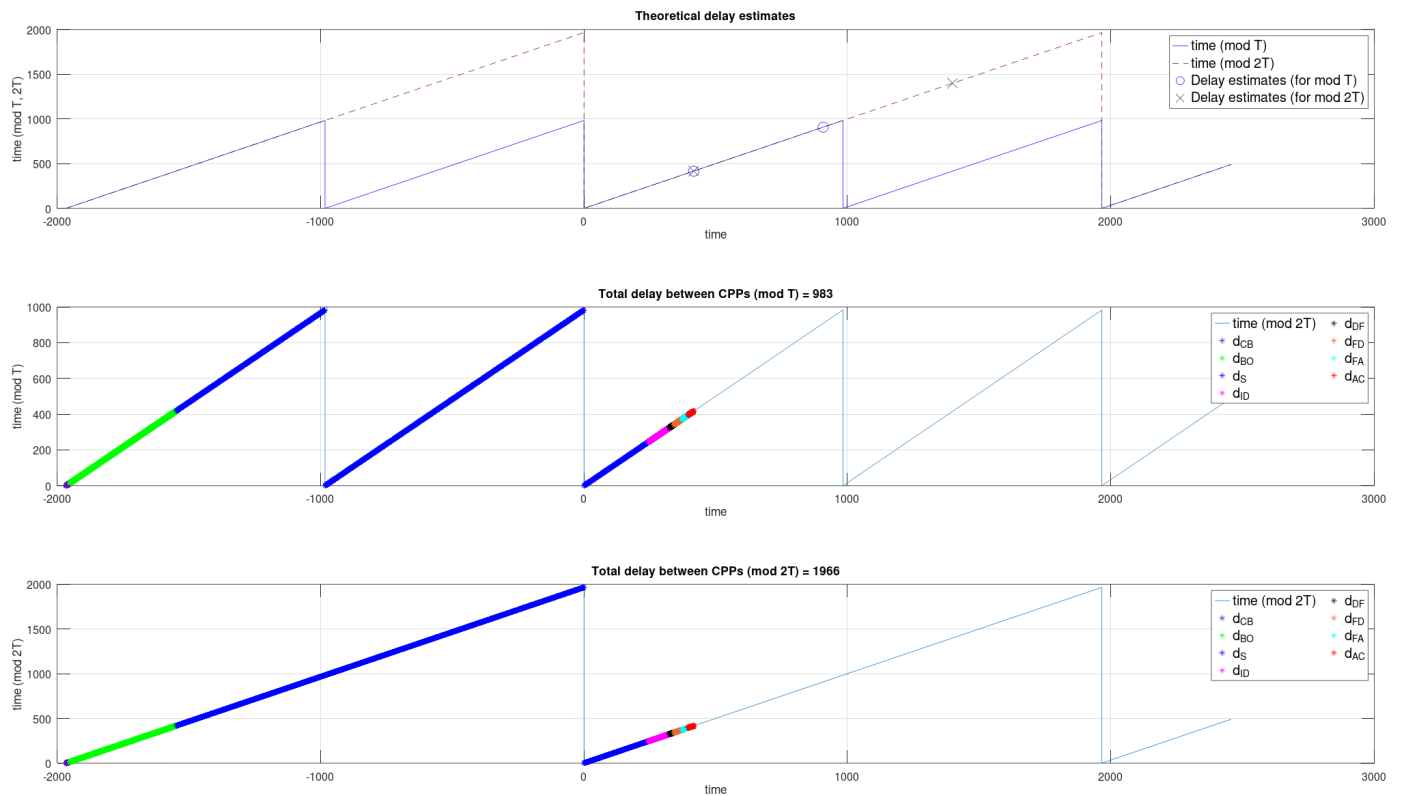


Figure 4. An example graphical representation of a mathematical simulation result.

3. Hardware Proof of Concept Implementation

The concept architecture is designed to be applicable to ASIC as well as FPGA designs. The latter offer lower cost and quicker development at the cost of less customizability. Initially, the feasibility of a successful implementation of the presented concept in an FPGA was questionable, e.g., due to the internal connection delay constraint given in Equation (2). To address and confirm or disprove these concerns, an FPGA was selected for the proof-of-concept design platform. Additionally, to verify the possibility of the whole concept implementation using FPGA chips only, no additional active components were used in the operated signal paths.

A commercially available Digilent Arty A7-35T [25] development board was selected as a target device for the concept implementation. The platform uses an AMD/Xilinx Artix 7 35T FPGA—specifically the XC7A35TICSG324-1L FPGA chip. This modern FPGA family is designed in a 28 nm process [26] and is a relevant example of the currently available FPGA technologies.

3.1. Pmod Adapter PCB

To ensure the best possible conditions for transmitting signals between development boards, a dedicated custom PCB adapter was designed. A 3D model of the PCB is presented in Figure 5. Its primary function is to transmit/receive clock signals used for phase alignment procedure using SMA connectors and traces with 50 Ω impedance. Additionally, FPGA LVCMOS drivers were configured to 8 mA drive strength for output impedance of approximately 50 Ω according to [27].

Unfortunately, the Arty A7 development board does not include termination resistors near the FPGA pins. Therefore, a Thevenin termination (suggested in [27]) had to be included on the additional Pmod adapter, which is not ideal from a signal integrity perspective. The actual transmission lines are composed of SMA cables, SMA to Pmod traces on the adapter PCB, and Arty A7 Pmod connector to FPGA pins traces. Because of this, the maximum supported signal frequency had to be verified by comparing the signal transmitted to and from the FPGA using an oscilloscope. The maximum reliable

operational frequency was established as 50 MHz. According to the timing report generated by Vivado, the implemented clock phase alignment system supported at least 400 MHz (constraints for higher frequencies were not verified). It should be noted that the maximum operational frequency is constrained by the signal integrity of the PCB, and this could be improved with a different hardware implementation, e.g., splitting the transmission lines using additional external buffers.

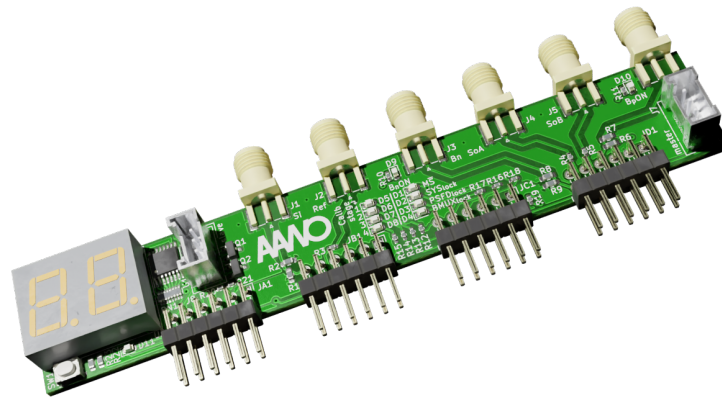


Figure 5. Three-dimensional model of the designed Pmod to SMA adapter PCB.

Additionally, the designed PCB adapter also includes connectors for an SPI communication interface between the connected nodes, additional LEDs, a button, and a two digit 7-segment display to monitor and control the status of operation.

3.2. Clock Forwarding

In general, FPGA chips include clock-capable input pins, but clock output pins are not specified by default. This limitation can be overcome using the clock forwarding technique described in [28]. It uses double data-rate (DDR) registers with constant and opposite data inputs. The clock signal, instead of being directly routed to the output buffer, connects to the DDR register's clock input. A schematic diagram of this concept is depicted in Figure 6.

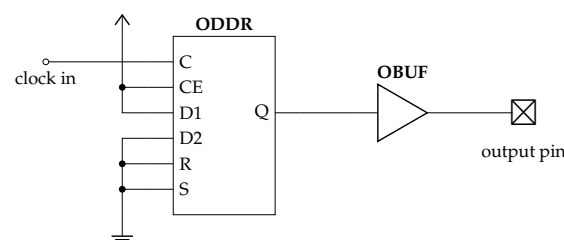


Figure 6. Diagram of the clock forwarding technique used in FPGA designs (adapted from [28]).

3.3. FPGA Implementation of Clock Phase Alignment System

In order to ensure the widest possible frequency range support, as well as the lowest additional jitter possible, the phase alignment system components (excluding the blocks responsible for the controlling functionalities) had to be implemented using FPGA dedicated clocking resources. This type of implementation removes the timing limitations typical to digital circuit design. In contrast, the usage of standard FPGA fabric and resources would require complicated correction circuits, similarly to delay generators [29].

The usage of FPGA dedicated clocking resources does not necessarily impose an additional operational frequency range limit over the actual FPGA components limits. In the case of the FPGA chip used for this work, the operational frequency is limited by the mixed-mode clock manager (MMCM) block and BUFG buffers. The input clock frequency of the former is in the range between 10 MHz and 800 MHz, while the maximum frequency of the latter is 464 MHz [30]. Practically, the minimum operational frequency

of the phase alignment system equals 20 MHz, because it utilizes a signal with two times lower frequency during the calibration procedure. Additionally, as previously stated, the adapter PCB with no additional buffers limited the actual maximum frequency of the proof-of-concept setup to 50 MHz.

Figure 7 depicts a simplified diagram of the clock phase alignment system using FPGA resources. For phase shifting functionality, the MMCM block (marked MMCM_A in the Figure 7) is used. It is configured such that the CLKOUT0 output port always replicates the input frequency and the CLKOUT1 port always outputs a two times lower frequency. These two signals are then multiplexed using a global clock mux buffer (BUFGMUX), effectively creating a selectable frequency divider. The clock outputs are implemented using the clock forwarding technique with bi-directional buffers (IOBUF) or a simple output buffer (OBUF), depending on the output pin. An additional reference output pin was implemented to monitor and compare the signals using an external oscilloscope. In order to minimize the signal jitter, the multiplexer from the concept diagram in Figure 3 was replaced with an additional MMCM (marked MMCM_B in Figure 7), utilizing both its clock inputs. This block is also always configured such that the CLKOUT0 output port replicates the input frequency.

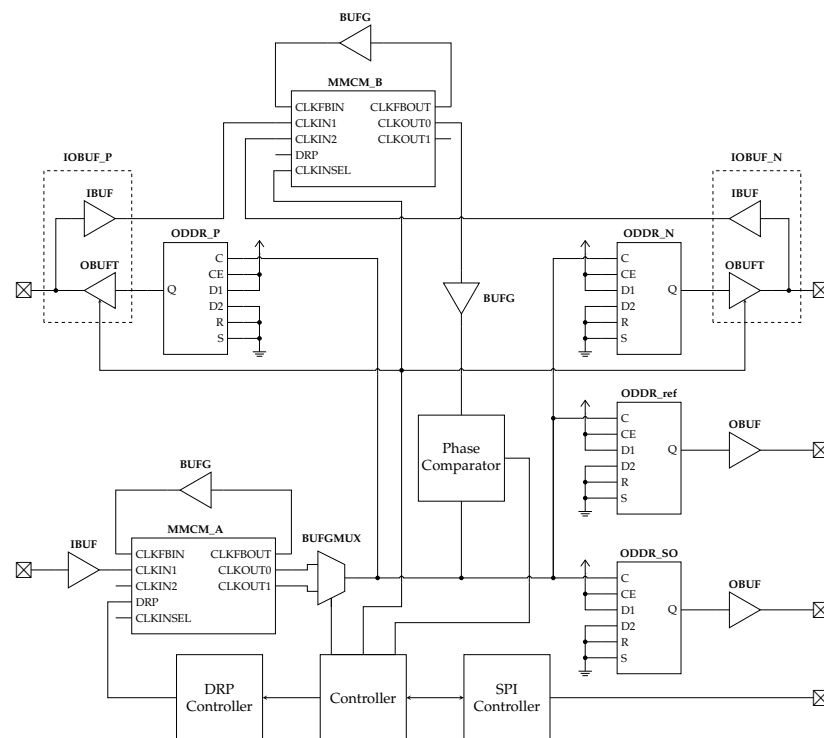


Figure 7. Simplified block diagram of a single clock phase alignment system implemented in FPGA.

3.4. MMCM Configurations

In 7 Series FPGAs, the MMCM blocks serve as jitter filters and frequency synthesizers for a wide range of frequencies. The internal phase-frequency detector (PFD) compares the phase and frequency of the rising edges of both the input (reference) clock and the feedback clock. The PFD is used to generate a signal that drives the charge pump (CP) and loop filter (LF) to generate a reference voltage to the voltage controlled oscillator (VCO). Each of the multiple VCO output phases can be selected as the reference clock to the output counters. Additionally, the MMCM contains a special counter controlling the feedback clock of the MMCM. As well as the phase shifts, each counter can be independently programmed [31].

The relationships between input, output, and VCO operating frequency are determined by the following equations [31]:

$$F_{VCO} = F_{CLKIN} \frac{M}{D} \quad (11)$$

$$F_{OUT} = F_{CLKIN} \frac{M}{D \cdot O} = \frac{F_{VCO}}{O} \quad (12)$$

where the values of M, D, and O counters correspond to

- M—a VCO clock multiplier, equal to CLKFBOUT_MULT_F setting.
- D—a division ratio for all output clocks, equal to DIVCLK_DIVIDE setting.
- O—a division ratio for individual clock output, equal to CLKOUT0_DIVIDE_F or CLKOUT[1:6]_DIVIDE settings.

The output clock phase shift in the MMCM is implemented using two mechanisms: coarse phase shifting and a delay time counter. The coarse phase shifting has a resolution of 45° relative to the VCO clock period and the delay time counter delays the output by a given number of VCO clock cycles. As a result, there is a direct correlation between the possible phase shift for the clock output, the VCO clock frequency, and the a division ratio for the individual output. The maximum delay time counter phase offset is 64 VCO clock cycles [32].

During the proof-of-concept verification, all MMCMs were configured to generate the maximum supported VCO frequency. In the case of the FPGA chip used for this work, the MMCM VCO frequency must be in range between 600 MHz and 1200 MHz [30]. As a result, the maximum phase shifting resolution is ensured. What is more, the MMCM operates within the specified VCO frequency range during calibration stages when the signal with two times lower frequency is used, without changing configuration parameters. A set of MMCM parameters that provide the described behavior are listed in Table 1.

Table 1. MMCM parameter values for input frequency replication with maximum VCO frequency (integer frequencies up to 50 MHz).

Input & Output Frequency	M ¹ Parameter Value	D ² Parameter Value	O ³ Parameter Value	Number of Phase Shift Steps	Theoretical Phase Shift Resolution
20 MHz	60	1	60	256	195.3125 ps
24 MHz	50	1	50	256	162.7604 ps
25 MHz	48	1	48	256	156.2500 ps
30 MHz	40	1	40	256	130.2083 ps
40 MHz	30	1	30	240	104.1667 ps
48 MHz	25	1	25	200	104.1667 ps
50 MHz	24	1	24	192	104.1667 ps

¹ M corresponds to the CLKFBOUT_MULT_F setting; ² D corresponds to the DIVCLK_DIVIDE setting; ³ O corresponds to the CLKOUT0_DIVIDE_F or CLKOUT[1:6]_DIVIDE setting.

The DRP (dynamic reconfiguration port) controller module pictured in Figure 7 is based on the Xilinx Application Note [32] and a corresponding reference design. It is responsible for configuring the MMCMs. The DRP controller was used to update the MMCM parameters before changing the clock signal frequency during verification, eliminating the need to reconfigure the FPGA chips to enable support for different MMCM input frequencies. The other usage for the DRP controller was to selectively phase shift the signals. A similar DRP controller (with no phase shifting capability) was connected to the MMCM_B block. To simplify the Figure 7, the latter DRP controller was omitted from the diagram, as it does not serve any role in the actual phase alignment system calibration.

As previously stated, the MMCM multiplier and division parameters are constant during the whole calibration procedure. However, these could be changed during calibration stages when the signal with two times lower frequency is used. This should enable the usage of any input signal frequency (compared to those listed in Table 1). To simplify the design, this option was omitted and could be another interesting topic for further research.

3.5. Satisfying the Internal Connection Delay Constraint

One of the biggest obstacles during implementation in the FPGA structure was ensuring that the internal connection delay constraint given in Equation (2) was satisfied with acceptable precision. Achieving this across all four PVT timing corners made this task even more difficult.

By using a combination of deliberate port selection, direct resource instantiation, and manual placement and routing of the phase alignment system components (excluding the controller block), the goal was finally achieved. These methods give a designer total control over the synthesis tool, prevent optimizations, and enable implementation of features not possible to implement otherwise [33]. Table 2 depicts the actual route delays reported by Xilinx Vivado for all timing corners.

Table 2. Internal connection delays reported by Xilinx Vivado with total values for internal connection constraints for four PVT corners.

Net Source	Net Target	FAST_MAX Corner	FAST_MIN Corner	SLOW_MAX Corner	SLOW_MIN Corner
MMCM_A.CLKOUT0	BUFGMUX.I0	534 ps	489 ps	1666 ps	1587 ps
MMCM_A.CLKOUT1	BUFGMUX.I1	534 ps	489 ps	1666 ps	1587 ps
BUFGMUX.out	ODDR_P.C	846 ps	576 ps	1649 ps	1507 ps
ODDR_P.Q	IOBUF_P.I	2 ps	2 ps	2 ps	2 ps
BUFGMUX.out	ODDR_N.C	847 ps	577 ps	1651 ps	1509 ps
ODDR_N.Q	IOBUF_N.I	2 ps	2 ps	2 ps	2 ps
IOBUF_P.O	MMCM_B.CLKIN1	480 ps	440 ps	1233 ps	1162 ps
IOBUF_N.O	MMCM_B.CLKIN2	480 ps	440 ps	1233 ps	1162 ps
$d_{AP} + d_{NM}$		1328 ps	1018 ps	2884 ps	2671 ps
$d_{BN} + d_{AC} + d_{CB} + d_{PM}$		1329 ps	1019 ps	2886 ps	2673 ps

It is worth emphasizing that the inaccuracy in satisfying the internal connection delay constraint varied from 1 ps to the maximum of 2 ps across all PVT corners.

3.6. Phase Comparator

The phase comparator module is based on the same architecture as the relative jitter measuring system presented in [34] and is depicted in Figure 8. The main part of the module consists of two symmetrical tapped delay lines (TDLs) implemented as carry chains in the FPGA. This carry chain implementation of TDLs is common in time to digital converter (TDC) designs, e.g., in [29,35–38]. In the case of this work, each TDL contained 256 taps. The output of each tap was connected to a synchronizer consisting of a latch and a flip-flop, synchronizing the delayed signal to a system clock domain. The reason for using a latch and flip-flop architecture over the typical two consecutive flip-flops was to reduce the effects of metastability, as described in [34]. The registered TDLs' taps effectively created a temporary snapshot of an input signal being transmitted through the delay line. The stored values were mutually compared using an array of XOR gates connected to the synchronizers' outputs. As a result, the output vector of the XOR gates was equivalent to the input signal misalignment.

The placement of the TDLs was manually selected so that the delays of both TDLs matched (across all taps) and the system clock source was matched to each corresponding flip-flop delay. All of the aforementioned delay requirements were precisely matched for all four process corners reported by Xilinx Vivado (with 1 ps precision). Until the very last interconnects to the flip-flops in the slices, the system clock signal was transmitted using the same path, as depicted in Figure 9. These results suggest that the selected logic cells and their interconnects are the closest implementation of two mirrored and symmetrical TDLs that is possible to achieve in this FPGA chip.

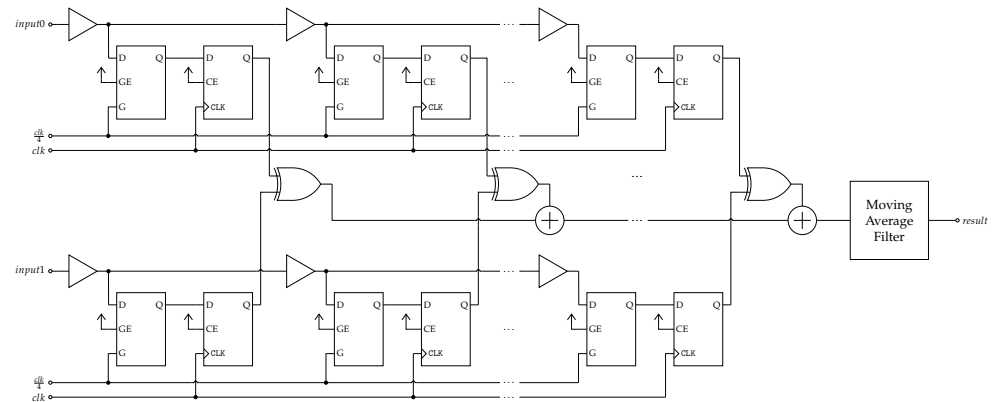


Figure 8. Simplified diagram of TDL-based phase comparator architecture.

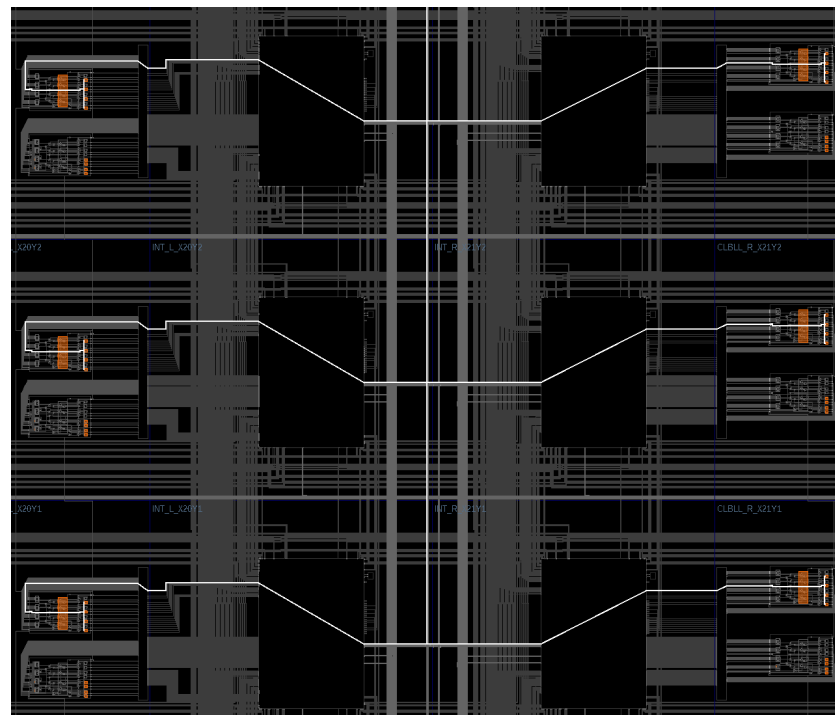


Figure 9. Selected section of an implemented device, illustrating the system clock to TDL flip-flops connection. The orange elements are parts of the TDLs.

A finite impulse response, moving average (MA) filter was used for smoothing the stream of XOR-ed differences. This type of filter was selected due to its sharper step response [39] compared to other types of filters. Additionally, it is quite simple in design as well as in implementation.

Unlike the design described in [34], the phase comparator does not contain a transition detection circuit. The module performance was verified using two outputs of the same MMCM block connected to two inputs of the phase comparator using similar nets of the same total delay reported by Vivado. Both MMCM output signals were always configured with the same multiplier and divider parameters, to generate the same frequency. One of the outputs was phase shifted in a 256-step range using a DRP controller—as described in Section 3.4.

Figure 10 presents a comparison of the phase comparator module's output range across different filter lengths and input clock signal frequencies. The tests of the module showed that filtering samples of XOR-ed differences with an adequate filter length produced a more linear and monotonic representation of the phase shift between the input signals. The worst performance phase offset detection was observed for the 50 MHz input clock signal

frequency—as depicted in the Figure 10d. However, the 65,536 filter length significantly improved the differentiability between phase shifts—especially close to 0° and 180° phase offsets. On the other hand, the range of MA filter output values for phase shifts close to 90° and 270° varied significantly, even for the 65,536 filter length. This effect was probably caused by the total TDL delay being comparable to the input signals' frequency. The phase comparator architecture is potentially an interesting topic for further research, especially including higher frequencies of compared signals.

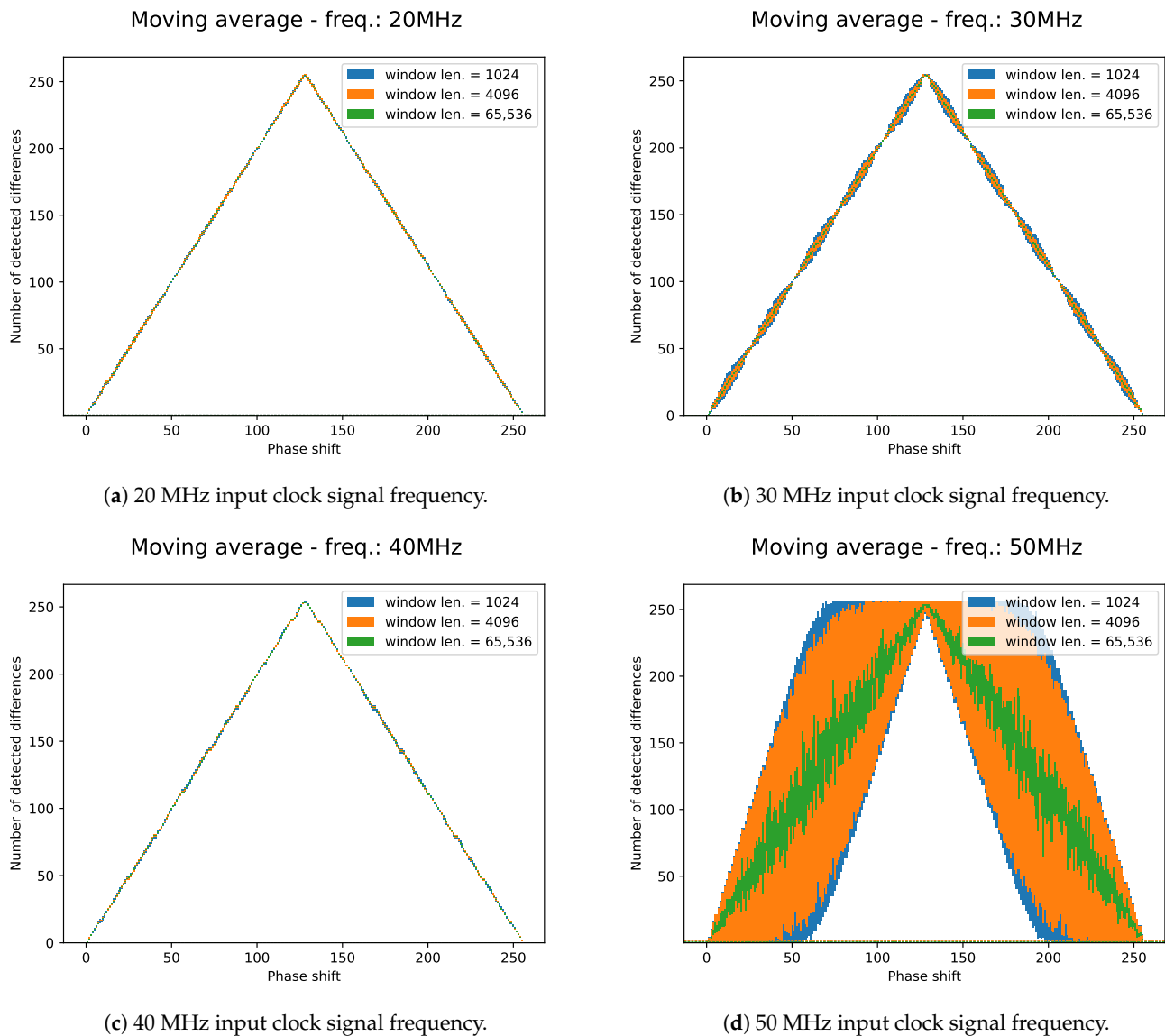


Figure 10. Comparison of phase comparator module performance for different input clock signal frequencies.

4. Verification and Measurement Methodology

The concept was verified using three separate instances of the Arty A7 board, each with a Pmod adapter PCB (described in Section 3.1) attached and connected using SMA cables for clock signals, as well as separate 4 wire cables for SPI bus. The primary clock signal used for synchronization was generated using a Si5351 clock generator. Its maximum output jitter equaled 100 ps peak-to-peak [40]. The system clock frequency was set to 50 MHz and was generated from an on-board oscillator. A simplified block diagram of a verification and measurement testbench setup is depicted in Figure 11. For each measured

case, the calibration procedure was performed once and measurements were started after its completion.

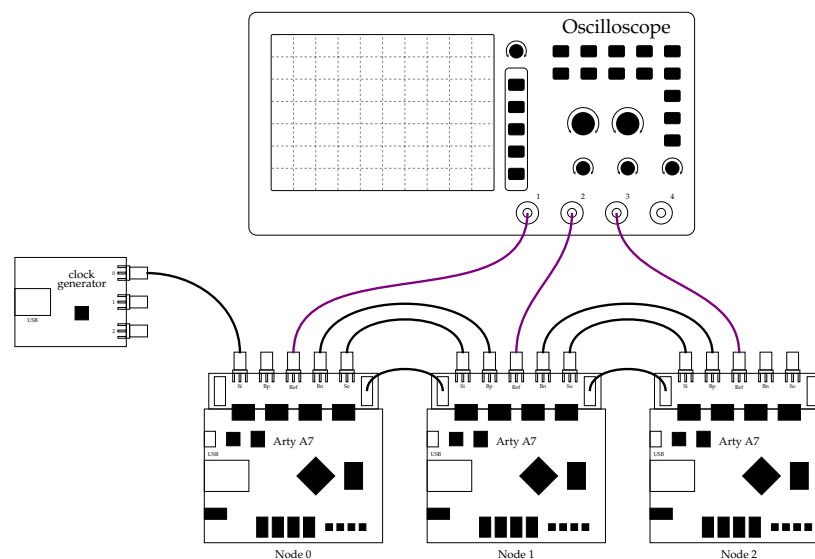


Figure 11. Simplified diagram of verification and measurement testbench setup.

As described in Section 3.3, the measurements were performed using an additional reference output (presented in Figure 7) connected to an external oscilloscope. Before the node calibration procedure and final phase shift measurements, the state of reference signals was observed to confirm the initial phase offset between each node.

Additionally, before the synchronization clock frequency change, the MMCM blocks' parameters were reconfigured using the DRP controllers (mentioned in Section 3.4). All of the utilized MMCMs parameters values are listed in Table 1. Each nodes' MMCM current multiplier and divider parameters are displayed on a two digit 7-segment display located on the adapter PCB of each node. This allows the user to monitor if the MMCMs are configured for the same input signal frequency.

5. Results

The proof-of-concept design was evaluated in multiple cases, using four different configurations of two 50 Ω SMA cables: 0.5 m and 1.0 m. Additionally, multiple frequencies of synchronizing clock signal were verified for each configuration. The difference in length, and therefore in signal delay, was easily noticeable in the oscilloscope waveform. Figure 12 depicts a waveform comparison of three node 20 MHz clock signals before and after synchronization. Combined, these various test conditions covered a wide range of potential cases. The same absolute length (and therefore propagation delay) of connection between the nodes resulted in different compensation values for different signal frequencies. Table 3 presents the obtained results for the phase offset (skew) between nodes after the calibration procedure. The presented measurements were averaged over at least 10,000 samples.

The measurements presented in Table 3 are comparable to the white rabbit project's results. The accuracy of the white rabbit project is the mean of the offset between the nodes' outputs, while the precision is the standard deviation of this offset [2]. Using the same measures, the presented work achieved sub-nanosecond accuracy and a hundreds of picoseconds precision of synchronization. These results are similar to the white rabbit's results. It should once again be noted, however, that this work achieved these results for significantly less complex applications and cases compared to the white rabbit protocol. Additionally, it is worth mentioning that the presented solution covers only clock signal phase alignment capability and not time synchronization like is provided in the white rabbit protocol.

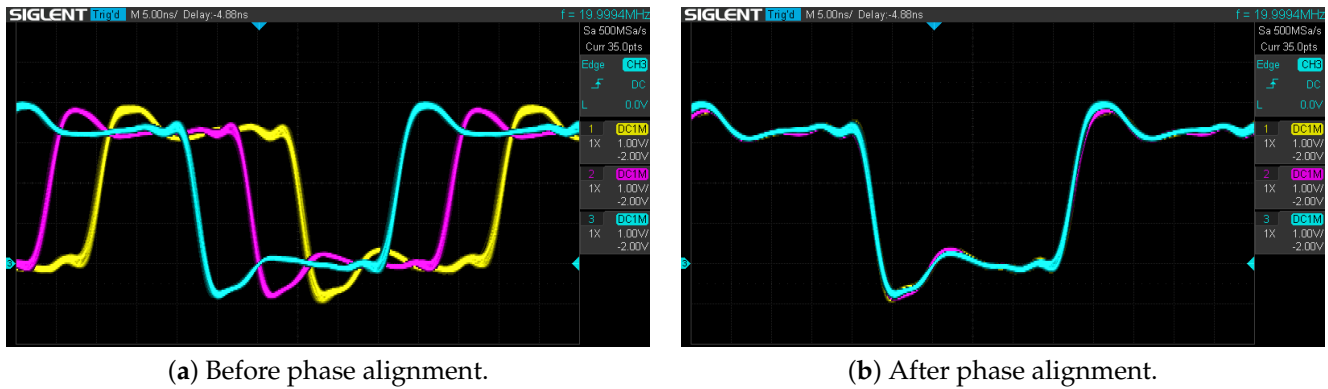


Figure 12. A sample waveform comparison of three nodes' reference signals before and after synchronization for the 20 MHz clock signal.

Table 3. Measured actual phase offset (skew) for multiple synchronizing clock signal frequencies and connection cable lengths.

Frequency	Unidirectional Connection Cable	Bidirectional Connection Cable	Measured Nodes 0–1 Phase Offset		Measured Nodes 0–2 Phase Offset	
			Mean	Standard Deviation	Mean	Standard Deviation
20 MHz	0.5 m	0.5 m	84.4 ps	91.1 ps	66.1 ps	81.1 ps
	0.5 m	1.0 m	30.3 ps	121.2 ps	212.1 ps	142.8 ps
	1.0 m	0.5 m	202.8 ps	124.4 ps	117.2 ps	103.2 ps
	1.0 m	1.0 m	108.9 ps	121.5 ps	50.2 ps	95.0 ps
30 MHz	0.5 m	0.5 m	98.3 ps	353.6 ps	219.8 ps	297.6 ps
	0.5 m	1.0 m	176.6 ps	180.8 ps	27.8 ps	189.5 ps
	1.0 m	0.5 m	105.1 ps	371.2 ps	46.5 ps	360.8 ps
	1.0 m	1.0 m	29.0 ps	263.8 ps	112.9 ps	276.2 ps
40 MHz	0.5 m	0.5 m	39.7 ps	245.2 ps	133.2 ps	268.9 ps
	0.5 m	1.0 m	106.4 ps	223.4 ps	102.6 ps	220.9 ps
	1.0 m	0.5 m	152.7 ps	272.2 ps	16.2 ps	262.5 ps
	1.0 m	1.0 m	223.5 ps	211.8 ps	38.9 ps	247.1 ps
50 MHz	0.5 m	0.5 m	312.3 ps	108.2 ps	364.9 ps	142.9 ps
	0.5 m	1.0 m	58.7 ps	208.6 ps	135.3 ps	188.1 ps
	1.0 m	0.5 m	112.7 ps	154.7 ps	15.3 ps	182.8 ps
	1.0 m	1.0 m	180.3 ps	160.1 ps	79.9 ps	175.9 ps

By comparing the measured mean phase offsets to the theoretical phase shift resolution for each input clock signal frequency presented in Table 1, it can be concluded that the maximum error in the final phase shift selection equaled 2 steps.

5.1. Comparison with Periodic Pulse Realignment

The presented work's precision and accuracy can be compared to the commonly used periodic realignment to a received synchronization pulse method described in Section 1. For accurate comparison and an example of a potential application, an additional 100 kHz pulse-width modulation (PWM) signal was generated in a 20 MHz clock domain in each node separately. A common periodic synchronization pulse was transferred to all nodes. The PWM signals' phase alignment was compared in three different cases:

- A separate 20 MHz clock domain in each node, derived independently from each node's external oscillator.
- A common, syntonized 20 MHz clock domain in each node, derived from the daisy-chained clock connection, with no phase alignment.
- A common, syntonized, and phase aligned 20 MHz clock domain in each node, derived from the daisy-chained clock connection.

In all cases, the 100 kHz PWM signal was phase aligned using a standard periodic pulse realignment method to a common synchronization pulse. Additionally, both the PWM signal and synchronization pulse were generated in the same clock domain. Figure 13 depicts the waveforms focused on 100 kHz PWM signal rising edges, as well as measurements for each aforementioned case. All waveforms were captured using a continuous oscilloscope trigger to visually display the difference between each type of synchronization method.

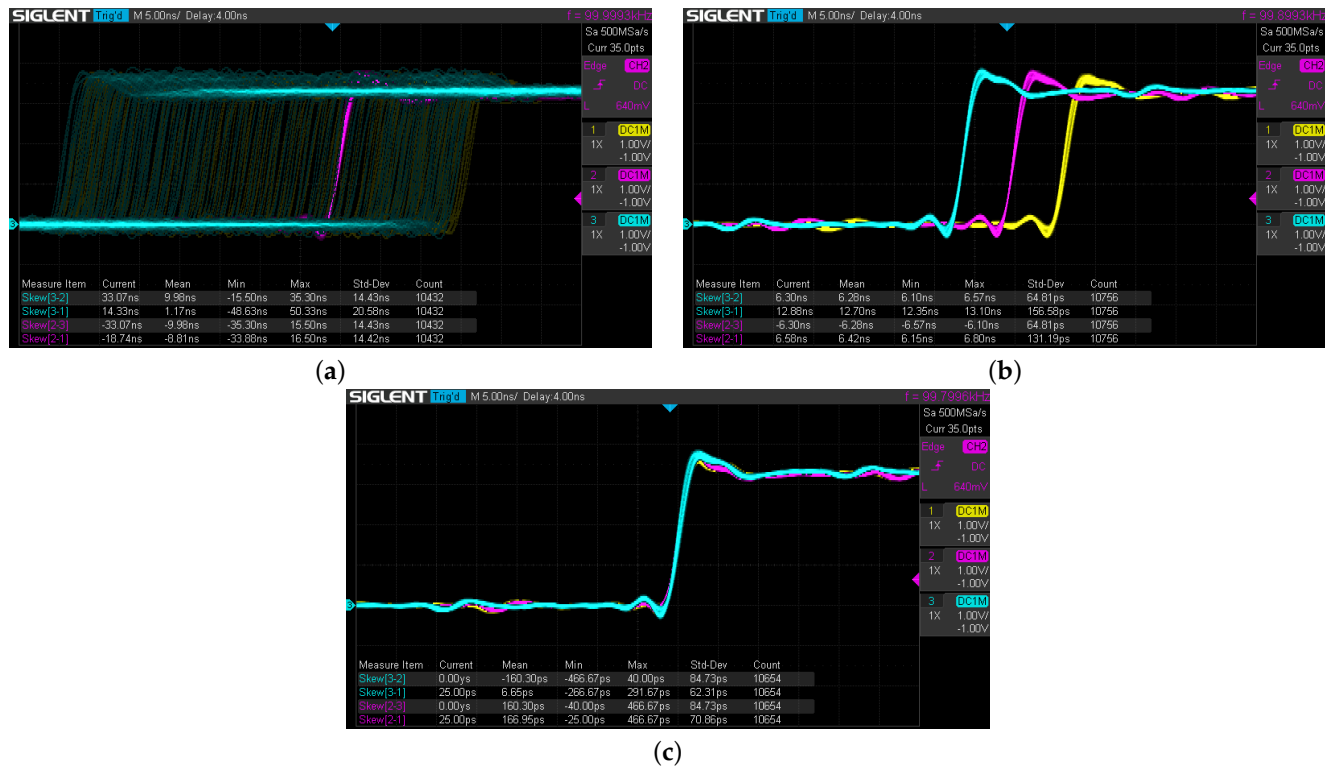


Figure 13. Waveform and skew measurement comparison of three types of synchronization of 100 kHz PWM signals generation across three separate nodes. (a) Standard periodic pulse realignment with PWM signals generated in separate 20 MHz clock domains; (b) PWM signals generated in a syntonized and not phase aligned 20 MHz clock domains across all nodes; (c) PWM signals generated in a syntonized and phase aligned 20 MHz clock domains across all nodes.

As expected, the periodic pulse realignment with independent clock domains achieved the worst results, with a phase alignment accuracy of approx. 9–10 ns and precision of approx. 14–15 ns. These results are below a 20 MHz clock period. The precision improved by 2–3 orders of magnitude using a syntonized 20 MHz clock domain with no phase-alignment for the 100 kHz PWM signal generation. The addition of clock domain phase alignment further improved the precision and accuracy of the synchronized PWM signal generation to tens of picoseconds and less than 200 ps, respectively.

Additionally, it is worth noting that the maximum value of the measured misalignment of the generated 100 kHz PWM signal equaled approx. 50 ns for the periodic pulse realignment with independent clock domains (as depicted in Figure 13a), compared to less than 500 ps for the syntonized and phase aligned 20 MHz clock domain (as depicted in Figure 13c).

The presented solution is more complex and requires noticeably more effort during implementation, compared to the standard periodic pulse realignment method. This is caused by the internal connection delay constraint given in Equation (2), which requires deliberate port selection, as well as manual placement and routing of several resources.

5.2. Resource Utilization

The clock signal phase alignment system proof-of-concept requires a small amount of the FPGA resources of the selected FPGA chip, even without any optimization. This is especially notable as the AMD/Xilinx Artix 7 35T is one of the least resourceful modern FPGA chips available on the market. Table 4 depicts the actual resource utilization of the implemented design. It shows that the presented design could be easily implemented on the AMD/Xilinx Artix 7 12T—the FPGA chip from the same family, containing the lowest amount of resources [41].

Table 4. Resource utilization of implemented proof-of-concept design.

Resource	Total Used	Available	Utilization [%]
LUT	3812	20,800	18.33
FF	2004	41,600	4.82
BRAM	18	50	36.00
DSP	0	90	0.00
MMCM	3	5	60.00

A detailed breakdown into subcomponents is presented in Table 5. The MMCM DRP controller module (with phase shifting capabilities) employs the highest amount of logic resources. This component can be relatively easily modified to implement a ROM with additional control, which would further reduce the logic utilization at the expense of BRAM usage. This could theoretically even enable the implementation of the presented design in the smallest available Spartan 7 FPGA [41]. As expected, the controller module required minimal FPGA resources. The phase alignment system and calibration algorithm (described in Section 2) are closely related and enable very efficient implementation of digital calculations and calibration control.

The presented proof-of-concept design includes SPI-based communication controllers (both master and slave) for data exchange between nodes during the calibration procedure. In a typical application, a communication bus (not necessarily SPI) between nodes would already be present and could be used for the phase alignment calibration. As previously described, the amount of data transmitted during calibration is minimal and there are no requirements in terms of transmission throughput, latency, etc. Practically any available communication method (e.g., APB, AXI, SPI, I²C, including custom interfaces) could be used for this task, by providing access to several control and status registers (CSRs) used by the calibration controller.

Furthermore, several of the components included in the proof-of-concept design (listed in Table 5) are either general and universally utilized in most (if not all) of the FPGA applications (such as `clk_rst_gen` or `debouncer` modules) or are purely for verification purposes (such as `SR_595_handler` or `seg7_CC_handler` modules). In practical applications, alongside the main set of functionalities in an FPGA chip, these modules would either already be implemented independently or not be required at all.

Additionally, in applications requiring a single, known in advance frequency of the synchronization clock signal, the `MMCM_B_DRP_ctrl` module would not be required, as no MMCM parameters would need to change. What is more, the `MMCM_A_DRP_ctrl` module resource utilization would be reduced significantly for the same reason. Only phase shifting capabilities would be required in this case.

In summary, the total resource utilization of the presented phase alignment system could be further reduced with no significant effort, depending on the application details. The general concept implementation also has space for optimization; however, the initial results are already very promising in this regard.

Table 5. Detailed resource utilization of implemented proof-of-concept design, divided into modules.

Module	LUT	FF	BRAM	DSP	MMCM
PhaseAlignSystem	438	1120	18	0	2
PhaseComparator	437	1120	18	0	0
DelayLine_A	0	512	0	0	0
DelayLine_B	0	512	0	0	0
MA_filter	56	84	18	0	0
Controller	353	179	0	0	0
Controller_master	344	161	0	0	0
Controller_slave	9	18	0	0	0
MMCM_A_DRP_ctrl	2645	165	0	0	0
MMCM_B_DRP_ctrl	55	55	0	0	0
Communication_ctrl	210	332	0	0	0
SPI_master	66	152	0	0	0
SPI_slave	17	38	0	0	0
clk_rst_gen	18	17	0	0	1
debouncer	15	14	0	0	0
SR_595_handler	18	23	0	0	0
seg7_CC_handler	70	94	0	0	0

5.3. Resource Utilization Comparison with the White Rabbit

Due to lack of other comparable examples, Table 6 presents comparison of this work's resource utilization with the white rabbit switch and node designs. Each compared design uses different FPGA chips, but the comparison is adequate as all of the compared FPGA platforms were designed by the same company (AMD/Xilinx), and include 6-input look-up tables (LUTs) that can be configured as two 5-input LUTs [42–45]. Except for the Spartan-6 FPGA that contains 18 kb Block RAM, which can be divided into two independent 9 kb BRAMs [42], all compared FPGA chips contain 36 kb Block RAM, which can be divided into two independent 18 kb BRAMs [43–45].

Compared to either of the white rabbit switch implementations, the presented design utilizes less than 5% of logic resources, less than 3% of flip-flops, and less than 7% of BRAM. Compared to the Spartan-6 based white rabbit node reference design, the presented proof-of-concept utilizes less than 32% of logic resources, less than 30% of flip-flops, and less than 63% of BRAM. However, the white rabbit system requires both WR switches and nodes in order to operate. In contrast, the presented phase alignment system utilizes the same synchronization design for all nodes.

Table 6. Resource utilization comparison with white rabbit switch and node.

Resource	This Work Artix-7 (XC7A35TICSG324-1L)	White Rabbit Switch (WRS) [46] Zynq UltraScale+ MPSoC (XCZU11EG-1FFVC1156E)	White Rabbit Switch (WRS) [47] Virtex-6 (XC6VLX240T)	White Rabbit Node Reference Design [48] Spartan-6 (XC6SLX45T-3FGG484)
LUT	3812	209,514	96k	8956
FF	2004	197,993	82k	6791
BRAM (18 k)	–	–	–	57.5 ¹
BRAM (36 k)	18 ²	517.5 ²	286 ²	–
DSP	0	3	no information	3
MMCM/PLL	3	3	no information	2

¹ Spartan-6 FPGA includes 18 kb BRAM that can be optionally programmed as two independent 9 kb BRAM [42];

² Zynq UltraScale+ MPSoC, 7 Series, and Virtex-6 FPGAs include 36 kb BRAM that can be optionally programmed as two independent 18 kb BRAM [43–45].

6. Conclusions

The functional verification of the presented hardware proof-of-concept design has shown that automatic clock signal phase alignment between daisy-chained nodes is possi-

ble without the use of a separate external control or monitor unit. It was also demonstrated that the phase alignment system can be implemented using only FPGA resources, with support for multiple frequencies of synchronization clock signal and with no additional external active components. Additionally, the obtained results proved that FPGA chips can successfully be used in applications requiring precise timing relations between individual components interconnects. However, further verification over long time periods and multiple FPGA chip units, as well as in different temperature conditions is required. Meticulous resource, as well as interconnecting route and delay, selection enables a relatively simple and effective implementation of a synchronization algorithm, which requires minimal resource utilization.

The presented clock signal phase alignment system's primary application is to enable the development of generators and sequencer-based systems that can be split into multiple separate blocks and cooperate with relatively high accuracy and precision. This could lead to the development of large systems constructed from multiple separate and phase-aligned devices. As mentioned in Section 1, example applications include sonar and radar systems, power generators, time-of-flight measurement systems, and more. The presented work was started as part of a research project of a multi-qubit quantum computer control system. Additionally, multiple clock generator circuits utilizing the presented concept could be functionally combined into a single generator, while retaining the ability to place individual ICs in different PCB locations. In large and sophisticated systems this could lead to PCB design simplification.

The presented proof-of-concept implementation contains several functionalities for general verification, such as MMCM frequency multipliers and divider parameter updating. These parts of the design would probably not be needed in a final application and could be removed, leading to even lower FPGA resource consumption by the clock signal phase alignment module.

The results measured after the phase alignment procedure presented in Table 3 indicated no correlation between the achieved skew between each node's reference signal and the synchronization clock signal frequency, nor the external connection length between nodes. The inaccuracies in the final phase alignment were most probably caused by the following effects:

- Zero phase offset detection precision in phase comparator module;
- Variations in internal interconnect delays comprising Equation (2) across adjacent nodes;
- Variations in buffer delays—specifically the reference signal output buffer, BUFGMUX and TSBs (all depicted in Figure 7);
- In some cases, the alignment error accumulation across the consecutive daisy-chained nodes.

The phase comparator module's precision was the most probable cause of the final phase alignment inaccuracies. Its performance could be improved by more research in this area or by changing the component's architecture. As mentioned in Section 2.6, essentially any phase detector design could be adapted to be used in the presented phase alignment system.

In an ideal case, the internal interconnect delay variations would be minimal. However, in the case of the presented FPGA proof-of-concept implementation, the theoretical maximum difference in interconnect delays in Equation (2) in the two separate FPGA chips, as presented in Table 2, equaled 1.868 ns, which is a significant value. The actual variations ought to be noticeably smaller, as the crucial interconnects share the exact same physical interconnects to the greatest extent possible. The results depicted in Table 3 showed that the actual interconnect delay variations were indeed definitely smaller.

The significant maximum interconnect delay difference was caused by the FPGA architecture. All FPGA chips use routing resources such as switch boxes. The FPGA routing architecture is the main culprit in making FPGAs worse than ASIC chips in area, delay, and power. A typical FPGA routing architecture uses about 70–90% of the total transistors on the die [49,50]. In the case of the selected Artix 7 35T FPGA, the routing resources did not support a direct connection from the MMCM block to the ODDR block in the same clock management tile (CMT). Despite the physical closeness of these two components, the

MMCM output signals had to be routed via a global buffer (BUFGMUX in the Figure 7). As a result, the internal clock phase alignment system's nets were implemented using an obligatory global buffer and several additional switch boxes. If the FPGA chip had switch boxes with the option to route a signal from the MMCM output to the ODDR block in the same CMT, it would be possible to significantly reduce the delay variation, as the metal interconnect sensitivity to variations is small compared with MOS transistors [51]. For the same reasons, this source of delay variation, and ultimately the inaccuracies in the final phase alignment, would be significantly reduced in an ASIC implementation.

The resource utilization of the initial non-optimized implementation enables the conclusion that the presented clock signal phase alignment module could be successfully implemented in most (if not all) modern FPGA chips. It can be included in a larger, more complex system as a separate low-end and relatively cheap FPGA device, or as a submodule of an existing design implemented in a larger FPGA. The latter would probably be the preferred option, as applications requiring or benefiting from the accuracy and precision of the presented results (i.e., compared to the periodic realignment method described in Section 1) already include at least one FPGA chip with a significant amount of resources, especially considering the fact that the presented concept does not enforce any additional requirements for PCB design (such as traces skew), apart from typical signal integrity aspects. However, having the ability to select a different architecture approach is beneficial from a system design perspective.

Additionally, the resource-efficient implementation suggests that the presented work could theoretically be implemented alongside a white rabbit node to enable obtaining the benefits of both solutions: precise timing synchronization across multiple bigger systems, as well as the ability to add additional separate precisely phase-aligned subsystems to the nodes. In effect, even more complex and sophisticated instruments and equipment could be designed by combining the two techniques.

Potential Further Research

In this work, the proof-of-concept design was demonstrated to work correctly with no external active components beside an FPGA chip. However, combined with the selected development board, it became apparent that this was a limiting factor of the maximum operational frequency of the verification setup. Therefore, the design ought to be verified with an improved revision of the PCB adapter or a different hardware setup, in order to measure the performance using higher frequencies of synchronization clock signal.

Another interesting research topic would be the verification of the stability of the phase alignment in different conditions, such as over a longer time period, with temperature variations or larger numbers of nodes. Additionally, concept development to support the tree topology together with daisy-chain topology might be beneficial as well.

The phase comparator module precision seemed to be a limiting factor in the final phase shift selection accuracy. Further research is planned in this area, to improve the phase offset detection and reduce the measurement latency caused by MA filter length, without significantly increasing resource consumption.

7. Patents

The presented clock phase synchronization system and algorithm are a part of a patent granted by The Patent Office of the Republic of Poland (UPRP): "Layout, system and method of precise clock phase synchronization. 4 October 2021, Application No. P.439121".

Funding: The APC was funded by the Institute of Microelectronics and Optoelectronics, Warsaw University of Technology.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

Acknowledgments: The author would like to thank Witold A. Pleskacz for APC funding acquisition and Krzysztof Marcinek and Witold A. Pleskacz for useful comments on and for proofreading an earlier draft of this manuscript.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

ASIC	Application-Specific Integrated Circuit
CPP	Common Phase Point
DDR	Double Data Rate
DRP	Dynamic Reconfiguration Port
FPGA	Field-Programmable Gate Array
LUT	look-up table
MA	Moving Average
MMCM	Mixed-Mode Clock Manager
PLL	Phase-Locked Loop
PTP	Precision Time Protocol
PVT	Process, Voltage, and Temperature
PWM	Pulse-width modulation
TDC	Time to Digital Converter
TDL	Tapped Delay Line
TSB	Tri-State Buffer
VCO	Voltage Controlled Oscillator
WR	White Rabbit

References

1. Rizzi, M.; Lipinski, M.; Wlostowski, T.; Serrano, J.; Daniluk, G.; Ferrari, P.; Rinaldi, S. White rabbit clock characteristics. In Proceedings of the 2016 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control, and Communication (ISPCS), Stockholm, Sweden, 3 September 2016; pp. 1–6. [\[CrossRef\]](#)
2. Lipiński, M.; van der Bij, E.; Serrano, J.; Wlostowski, T.; Daniluk, G.; Wujek, A.; Rizzi, M.; Lampridis, D. White Rabbit Applications and Enhancements. In Proceedings of the 2018 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control, and Communication (ISPCS), Geneva, Switzerland, 30 September–5 October 2018; pp. 1–7. [\[CrossRef\]](#)
3. Packi, F.; Beutler, F.; Hanebeck, U.D. Wireless acoustic tracking for extended range telepresence. In Proceedings of the 2010 International Conference on Indoor Positioning and Indoor Navigation, Zurich, Switzerland, 15–17 September 2010; pp. 1–9. [\[CrossRef\]](#)
4. Han, Y.; Zhu, Y. Implementation of a VI-Based Synchronic Testing System for Underwater Transducer Array. In Proceedings of the 2011 International Conference on Network Computing and Information Security, Guilin, China, 14–15 May 2011; Volume 2, pp. 363–366. [\[CrossRef\]](#)
5. Baba, S.; Gajewski, W.; Jasiński, M.; Żelechowski, M.; Kaźmierkowski, M.P. High Performance Power Supplies for Plasma Materials Processing. *IEEE Access* **2021**, *9*, 19327–19344. [\[CrossRef\]](#)
6. Hovhannisyanyan, B.; Margaryan, N.; Tsaturyan, G.; Antonyan, S.; Ohanyan, G.; Manvelyan, M. Method for precise synchronization between multiple vector signal generators. *Proc. YSU A Phys. Math. Sci.* **2020**, *54*, 61–64. [\[CrossRef\]](#)
7. Salarpour, M.; Farzaneh, F.; Staszewski, R.B. Synchronization-Phase Alignment of All-Digital Phase-Locked Loop Chips for a 60-GHz MIMO Transmitter and Evaluation of Phase Noise Effects. *IEEE Trans. Microw. Theory Tech.* **2019**, *67*, 3187–3199. [\[CrossRef\]](#)
8. Serrano, J.; Alvarez, P.; Cattin, M.; Garcia Cota, E.; Lewis, J.; Moreira, P.; Wlostowski, T.; Gaderer, G.; Loschmidt, P.; Dedič, J.; et al. The White Rabbit project. In Proceedings of the ICALEPCS2009 Kobe, Japan, 12–16 October 2009; pp. 93–95.
9. Yang, Z.; Ma, Y.; Yang, W.; Zhang, Y. Implementation of White Rabbit Time Synchronization System in State Acquisition System of High-energy Physics Experimental Device. *J. Phys. Conf. Ser.* **2022**, *2264*, 012014. [\[CrossRef\]](#)
10. Derviškić, A.; Razzaghi, R.; Walger, Q.; Paolone, M. The White Rabbit Time Synchronization Protocol for Synchrophasor Networks. *IEEE Trans. Smart Grid* **2020**, *11*, 726–738. [\[CrossRef\]](#)
11. National Instruments. NI-TClk Overview. Available online: https://www.ni.com/docs/en-US/bundle/ni-tclk/page/nitclk/nitclk_overview.html (accessed on 15 March 2024).
12. Rizzi, M.; Lipinski, M.; Ferrari, P.; Rinaldi, S.; Flammini, A. White Rabbit Clock Synchronization: Ultimate Limits on Close-In Phase Noise and Short-Term Stability Due to FPGA Implementation. *IEEE Trans. Ultrason. Ferroelectr. Freq. Control* **2018**, *65*, 1726–1737. [\[CrossRef\]](#) [\[PubMed\]](#)
13. Lee, S.; Kim, J.; Choi, K.; Ryu, S. Method and Device to Align Phases of Clock Signals. U.S. Patent 2017250695, 31 August 2017.

14. Jun'an, Z.; Donging, F.; Ruitao, Z.; Jun, L.; Yujun, Y.; Pu, L.; Xiamjie, W.; Guangjun, L. Multichip Synchronization Structure Based on Time-Digital Converter Circuit. China Patent CN106970679, 21 July 2017.
15. Kuddes, D.W. Method and System for Aligning the Phase of High Speed Clocks in Telecommunications Systems. U.S. Patent US5638410A, 10 June 1997.
16. Wojciechowski, A.A.; Marcinek, K.; Pleskacz, W.A. Clock Signal Phase Alignment System for Daisy Chained Integrated Circuits. In Proceedings of the 2022 29th International Conference on Mixed Design of Integrated Circuits and System (MIXDES), Wrocław, Poland, 23–24 June 2022; pp. 89–92. [\[CrossRef\]](#)
17. Jones, G.A.; Jones, J.M. *Elementary Number Theory*; Springer Undergraduate Mathematics Series; Springer: London, UK, 1999.
18. Mestice, M.; Ciarpì, G.; Rossi, D.; Saponara, S. An Integrated Charge Pump for Phase-Locked Loop Applications in Harsh Environments. *Electronics* **2024**, *13*, 744. [\[CrossRef\]](#)
19. Park, K.; Shim, M.; Ko, H.G.; Nikolić, B.; Jeong, D.K. Design Techniques for a 6.4–32-Gb/s 0.96-pJ/b Continuous-Rate CDR With Stochastic Frequency–Phase Detector. *IEEE J.-Solid-State Circuits* **2022**, *57*, 573–585. [\[CrossRef\]](#)
20. Markoni, F.; Tiyono. Implementation XOR Logic Gate for Phase Difference Detector in Automatic Synchronizer for Synchronous Generator. In Proceedings of the 2021 International Conference on Technology and Policy in Energy and Electric Power (ICT-PEP), Jakarta, Indonesia, 29–30 September 2021; pp. 33–36. [\[CrossRef\]](#)
21. Allan, D.; Daams, H. Picosecond Time Difference Measurement System. In Proceedings of the 29th Annual Symposium on Frequency Control, Atlantic City, NJ, USA, 28–30 May 1975; pp. 404–411. [\[CrossRef\]](#)
22. Moreira, P.; Alvarez, P.; Serrano, J.; Darwezeh, I.; Wlostowski, T. Digital dual mixer time difference for sub-nanosecond time synchronization in Ethernet. In Proceedings of the 2010 IEEE International Frequency Control Symposium, Newport Beach, CA, USA, 1–4 June 2010; pp. 449–453. [\[CrossRef\]](#)
23. Wojciechowski, A.A.; Marcinek, K.; Pleskacz, W.A. Dual TDL Based Phase Difference Detector Architecture. In Proceedings of the 2023 30th International Conference on Mixed Design of Integrated Circuits and System (MIXDES), Krakow, Poland, 29–30 June 2023; pp. 122–126. [\[CrossRef\]](#)
24. Eaton, J.W.; Bateman, D.; Hauberg, S.; Wehbring, R. *GNU Octave Version 9.1.0 Manual: A High-Level Interactive Language for Numerical Computations*; 2024. Available online: <https://github.com/gnu-octave/octave/blob/default/CITATION> (accessed on 3 July 2024).
25. Digilent. Arty A7. Available online: <https://digilent.com/reference/programmable-logic/arty-a7/start> (accessed on 5 July 2023).
26. Xilinx Inc. *Extending 28nm Leadership with an Expanded Portfolio and Lower Power*; Xilinx Inc.: San Jose, CA, USA, 2015.
27. Xilinx Inc. *7 Series FPGAs PCB Design Guide (UG483)*, v1.14 ed.; Xilinx Inc.: San Jose, CA, USA, 2019.
28. Xilinx Inc. *Using Constraints (UG903)*, v2022.2 ed.; Xilinx Inc.: San Jose, CA, USA, 2022.
29. Zhu, M.; Cui, T.; Qi, X.; Gao, Q. A Picosecond Delay Generator Optimized by Layout and Routing Based on FPGA. *Sensors* **2023**, *23*, 6144. [\[CrossRef\]](#) [\[PubMed\]](#)
30. Xilinx Inc. *Artix-7 FPGAs Data Sheet: DC and AC Switching Characteristics (DS181)*, v1.27 ed.; Xilinx Inc.: San Jose, CA, USA, 2022.
31. Xilinx Inc. *7 Series FPGAs Clocking Resources User Guide (UG483)*, v1.14 ed.; Xilinx Inc.: San Jose, CA, USA, 2018.
32. Tatsukawa, J. *MMCM and PLL Dynamic Reconfiguration (XAPP888)*, v1.8 ed.; Xilinx Inc.: San Jose, CA, USA, 2019.
33. Xilinx Inc. *UltraFast Design Methodology Guide for FPGAs and SoCs (UG949)*, v2022.2 ed.; Xilinx Inc.: San Jose, CA, USA, 2022.
34. Wojciechowski, A.A.; Marcinek, K.; Pleskacz, W.A. Relative Jitter Measurement Methodology and Comparison of Clocking Resources Jitter in Artix 7 FPGA. *Electronics* **2023**, *12*, 4297. [\[CrossRef\]](#)
35. Szplet, R.; Czuba, A. Two-Stage Clock-Free Time-to-Digital Converter Based on Vernier and Tapped Delay Lines in FPGA Device. *Electronics* **2021**, *10*, 2190. [\[CrossRef\]](#)
36. Sui, T.; Zhao, Z.; Xie, S.; Xie, Y.; Zhao, Y.; Huang, Q.; Xu, J.; Peng, Q. A 2.3-ps RMS Resolution Time-to-Digital Converter Implemented in a Low-Cost Cyclone V FPGA. *IEEE Trans. Instrum. Meas.* **2019**, *68*, 3647–3660. [\[CrossRef\]](#) [\[PubMed\]](#)
37. Dikopoulos, E.; Birbas, M.; Birbas, A. An Adaptive Downsampling FPGA-Based TDC Implementation for Time Measurement Improvement. *Chips* **2022**, *1*, 175–190. [\[CrossRef\]](#)
38. Tontini, A.; Gasparini, L.; Pancheri, L.; Passerone, R. Design and Characterization of a Low-Cost FPGA-Based TDC. *IEEE Trans. Nucl. Sci.* **2018**, *65*, 680–690. [\[CrossRef\]](#)
39. Al-Mbaideen, A. Application of Moving Average Filter for the Quantitative Analysis of the NIR Spectra. *J. Anal. Chem.* **2019**, *74*, 686–692. [\[CrossRef\]](#)
40. Silicon Laboratories. *Si5351A/B/C*; Silicon Laboratories: Austin, TX, USA, 2011.
41. Advanced Micro Devices, Inc. *AMD Cost-Optimized Portfolio Product Selection Guide (XMP100)*, v2.3 ed.; Advanced Micro Devices, Inc.: Santa Clara, CA, USA, 2024.
42. Xilinx Inc. *Spartan-6 Family Overview (DS160)*, v2.00 ed.; Xilinx Inc.: San Jose, CA, USA, 2011.
43. Xilinx Inc. *7 Series FPGAs Data Sheet: Overview (DS180)*, v2.6.1 ed.; Xilinx Inc.: San Jose, CA, USA, 2020.
44. Xilinx Inc. *Virtex-6 Family Overview (DS150)*, v2.5 ed.; Xilinx Inc.: San Jose, CA, USA, 2015.
45. Xilinx Inc. *Zynq UltraScale+ MPSoC Data Sheet: Overview (DS891)*, v1.10 ed.; Xilinx Inc.: San Jose, CA, USA, 2022.
46. Gumiński, M. *WRS Resource Utilisation on Xilinx US+ FPGA*; Technical report; Creotech Instruments S.A.: Piaseczno, Poland, 2019.
47. Daniluk, G. Resource Evaluation of WR Switch HDL for Ultrascale Plus. Available online: <https://ohwr.org/project/wr-switch-hdl-usp-eval/wikis/home> (accessed on 28 March 2024).

48. Florin, D. White Rabbit Node Reference Design. Available online: <https://ohwr.org/project/white-rabbit/wikis/WRReferenceDesign> (accessed on 28 March 2024).
49. Dehon, A. *Reconfigurable Architectures for General-Purpose Computing*; Technical report; Massachusetts Institute of Technology, Artificial Intelligence Laboratory: Cambridge, MA, USA, 1996.
50. Sivaswamy, S.; Wang, G.; Ababei, C.; Bazargan, K.; Kastner, R.; Bozorgzadeh, E. HARP: Hard-wired routing pattern FPGAs. In Proceedings of the 2005 ACM/SIGDA 13th International Symposium on Field-Programmable Gate Arrays, Monterey, CA, USA, 20–22 February 2005; pp. 21–29. [[CrossRef](#)]
51. Masuda, H.; Okawa, S.; Aoki, M. Approach for physical design in sub-100 nm era. In Proceedings of the 2005 IEEE International Symposium on Circuits and Systems (ISCAS), Kobe, Japan, 23–26 May 2005; Volume 6, pp. 5934–5937. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.